The AAAI-17 Workshop on
Knowledge-Based Techniques for Problem Solving and Reasoning
WS-17-12

Initial State Prediction in Planning

Senka Krivic,[1] Michael Cashmore,[2] Bram Ridder,[2]
Daniele Magazzeni,[2] Sandor Szedmak,[3] Justus Piater[1]

[1]Department of Computer Science, University of Innsbruck, Austria
firstname.lastname@uibk.ac.at

[2] Department of Computer Science, King's College London, United Kingdom
firstname.lastname@kcl.ac.uk

[3] Department of Computer Science, Aalto University, Finland
sandor.szedmak@aalto.fi

Abstract

While recent advances in offline reasoning techniques and online execution strategies have made planning under uncertainty more robust, the application of plans in partially-known environments is still a difficult and important topic. In this paper we present an approach for predicting new information about a partially-known initial state, represented as a multigraph utilizing Maximum-Margin Multi-Valued Regression. We evaluate this approach in four different domains, demonstrating high recall and accuracy.

1 Introduction

Planning in many domains means planning with incomplete and uncertain information. In such domains plans generated can be fragile. Contingency planning (Bonet and Geffner 2000; Hoffmann and Brafman 2005), conformant planning (Smith and Weld 1998; Palacios and Geffner 2006), and replanning techniques (Brafman and Shani 2014) work to make execution more robust, for an acceptable cost of computational difficulty or plan quality.

In this paper we focus on the problem of planning with incomplete information on the initial state in a deterministic domain. We describe a preprocessing step that predicts new information about a partially-known initial state inspired by research on *associative learning* (Hill 1984).

Humans associate certain holidays with specific sounds and smells, or foods with specific flavours, colours and textures. Pavlov and Anrep (Pavlov and Anrep 2003) have shown that it is possible to pair an unconditioned stimulus with another previously neutral stimulus. This makes it possible to learn and predict events in terms of associations between stimuli, representing the paradigm of *Classical conditioning*.

The idea is related to that of *assumption-based planning* (ABP) (Sammy Davis-Mendelow 2013), which formalises the idea of planning in a partially unknown initial state with assumptions. Davis-Mendelow et al. show that ABP has great utility for tasks that share a computational core with planning, and encapsulates a compelling for of common sense planning.

In this paper we present a novel way in which to make such assumptions, focusing on domains in which a robotic agent has to make fast decisions. For example scenarios such as the robocub where the robot has a partial information of the initial state and has to make fast decisions, or an autonomous underwater vehicle that must make critical decisions before its position drifts.

We demonstrate that it is possible to predict missing information in an initial state by exploiting the similarities among known facts. We pose the problem of prediction as that of learning missing edges in a graph. The learned edges are analogous to assumptions about facts within the initial state. To solve this associative learning problem we use a kernel-based approach for learning missing edges in a partially-given multigraph (Krivic et al. 2015). This allows us to propagate knowledge from existing relations to unknown relations in a partially-known initial state.

Maximum Margin Multi-Valued Regression (M^3VR) (Ghazanfar, Prügel-Bennett, and Szedmak 2012; Szedmak, Ugur, and Piater 2014) is applied to the class of learning problems where item-item relations might be given by different attributes. This learning framework is used at the core of a recommender system (Ghazanfar, Prügel-Bennett, and Szedmak 2012) and for predicting the effects of an action on pairs of objects in an affordance learning problem (Szedmak, Ugur, and Piater 2014). Their results show that it can deal with sparse, incomplete and noisy information. The M^3VR is compared with the state-of-the-art methods and is an established and competitive method for prediction of missing relations and recommender based systems (Ghazanfar, Prügel-Bennett, and Szedmak 2012; Krivic et al. 2015).

Krivic et al. (2015) use this method to refine a world model for planning to tidy up a child's room with a robotic agent. The system is used to learn the edges describing the possible spatial relations between objects. We build on this, describing how the approach can be generalised for use in any planning domain, learning edges that correspond to generic propositions in the initial state. In particular we describe: how the initial state is represented as a partially-known multigraph; the approach for learning missing edges; how the results of the learning framework are translated back into the planning domain; and finally, how they can be used in the planning process.

The problem is similar to a restricted class of contingency planning problems with deterministic actions, *partially observable Markov decision processes* (POMDP) (Bonet 2009). Techniques exist to solve these problems by first converting them into classical planning problems: K-Planner (Bonet and Geffner 2011), PO-RPR (Muise, Belle, and McIlraith 2014), and CLG (Albore and Geffner 2009); or using conformant planning techniques (Smith and Weld 1998; Palacios and Geffner 2006). Our work differs in that we do not translate the whole problem, but instead remove uncertainty by making predictions. In our problem there is no known probability distribution over the existence of propositions in the partially-known initial state. Moreover, the approach is orthogonal in that uncertainty is removed from the problem through prediction, either enabling a deterministic solution, or simplifying the contingency planning problem should they be used in combination.

By integrating the learning framework into a planning and execution system we demonstrate its efficacy on several domains. We perform an empirical evaluation on a range of problems in these domains. We show that:

- With already 20% knowledge of the initial state the accuracy of complete initial state prediction is 90%.
- Prediction leads to plans that are more robust (fewer replans are required) compared to an optimistic classical planning approach.
- Comparing with CLG, prediction increases the scalability of contingency planning, at a cost to robustness.

In Section 2 we describe our problem formulation for learning new relations in partially-known initial states. We describe the learning problem in Section 3, and explain how the learning framework is used to solve the formulated problem. In Section 4 we perform an evaluation. We conclude in Section 5.

2 Predictions in the Planning Problem

In this section we describe in detail our preprocessing step, which predicts new information about a partially-known initial state.

Definition 1 (Planning Problem) *A planning instance* Π *is a pair* $\langle Dom, Prob\rangle$*, where* $Dom = \langle Ps, As, arity\rangle$ *is a tuple consisting of a finite set of predicate symbols* Ps*, a finite set of (durative) actions As, and a function arity mapping all symbols in* Ps *to their respective arities. The triple* $Prob = \langle Ob, Init, G\rangle$ *consists of a finite set of domain objects Ob, the partial initial state Init, and the goal specification G.*

The atoms of the planning instance are the (finitely many) expressions formed by grounding – applying the predicate symbols Ps to the objects in Ob (respecting arities). The resultant expressions are the set of propositions P.

A state s is described by a set of literals formed from the propositions in P, $\{l_p, \neg l_p, \forall p \in P\}$. If every proposition from P is represented by a literal in the state, then we say that s is a *complete state*. A *partial state* is a set of literals $s' \subset s$, where s is a complete state.

The initial state $init$ is a partial state. A partial state can be *extended* into a complete state.

Definition 2 (Extending a Partial State) *Let* s' *be a partial state of Planning problem* Π*. Extending the state* s' *is a function* $Extend(\Pi, s') : s' \rightarrow s$ *where* s *is a complete state and* $s' \subset s$*.*

We describe a pre-processing step implementing $Extend$. All unknown propositional values in a partially-known initial state are predicted, producing a complete initial state. Briefly, the function $Extend(\Pi, s')$ is implemented as follows: the initial state $init$ is converted into a multigraph; edges in the multigraph are learned using M^3VR ; then the new edges are added as literals to the initial state.

First we describe the construction of the multigraph, then in Section 3 we describe the relational problem, and then how the learned relations are inserted back into the initial state. Finally, the complete initial state can be used by a classical planner to generate a plan.

Constructing the Multigraph

We represent a partially-known initial state $init$ as a partially-known multigraph M.

Definition 3 (Partially-known Multigraph) *A partially-known Multigraph* M *is a pair* $\langle V, E'\rangle$*, where* V *is a set of vertices, and* E' *a set of values of labelled, directed edges.*

The values assigned to all possible edges are $\{0, 1, ?\}$ corresponding to $\{$*not-existing*, *existing*, *unknown*$\}$. We use E' to denote a set of edges values in a partially-known multigraph, while E denotes the set of edges values in a completed multigraph. The partial state *init* is described as a partially-known multigraph with an edge for each proposition $p \in P$ that is either unknown or known to be true. That is:

$$\begin{aligned} V &\equiv Ob \\ E' &= \{e_{pred}(b,u) | (b,u) \in V \times V\} \end{aligned}$$

The existence of a *directed edge* $e_{pred}(b, u)$ between two vertices b and u for a predicate $pred$ is described by the function $L_{pred} : V \times V \rightarrow \{0, 1, ?\}$. For example, let b and u be two vertices in set V. For proposition p involving objects b and u, $L_{pred}(b, u) = 0$ if $\neg l_p \in init$, $L_{pred}(b, u) = 1$ if $l_p \in init$, and $L_{pred}(b, u) =?$ otherwise. Known edges are denoted with solid lines and unknown ones with dashed line such as in Figure 1. Edges are directed in the order the object symbols appear in the proposition. In the following we use B and U to differentiate between vertices of outgoing and incoming edges respectively. In our problem $B = U = V$.

Example

Consider the planning problem in figures 2 and 3. The problem describes a robot that is able to move between waypoints, pickup and manipulate objects, and put them in boxes. The predicates: `can-pickup`, `can-push`, `can-stack-on`, `can-fit-inside` describe whether it is possible to perform certain actions upon objects in the environment. In this problem we restrict our attention to four objects: `robot`, `cup01`, `box01`, and `block01`. In PDDL (Fox and Long 2003) literals that are absent from the initial

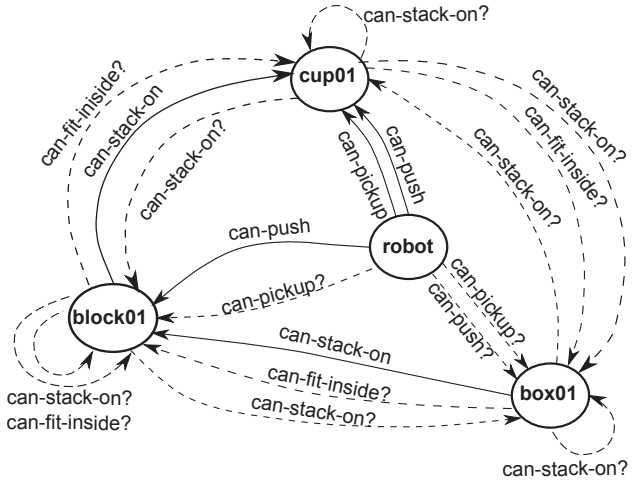

Figure 1: The graph M representing the initial state in the example problem. Solid edges correspond to propositions known to be true, $L_{pred}(b, u) = 1$. Dashed edges correspond to propositions whose value is unknown in the initial state, $L_{pred}(b, u) = ?$.

```
(define (domain toy-domain)
(:requirements :strips :typing ...)
(:types
  waypoint
  vehicle
  interactable
  box toy - interactable
  block - toy
  gripper)

(:predicates
  (connected ?wp1 ?wp2 - waypoint)
  (at ?v - vehicle ?wp - waypoint)
  (near ?v - vehicle ?wp - waypoint)

  (can-pickup ?r ?interactable)
  (can-push ?r ?interactable)
  (can-stack-on ?interactable ?interactable)
  (can-fit-inside ?interactable ?box)
  ...)

(:durative-action goto ...
(:durative-action pickup ...
(:durative-action putdown ...
(:durative-action stack ...
(:durative-action unstack ...
(:durative-action put_in_box ...
)
```

Figure 2: A fragment of the `toy-domain`. Some predicates and the body of operators are omitted for space.

state are assumed to be false. However, in this initial state those literals are assumed to be unknown.

A graph M is generated, the vertices of which are $Ob := \{robot, cup01, box01, block01\}$. The graph is shown in figure 1.

```
(define (problem toy-example-problem)
(:domain toy-domain)
(:objects
  wp1 wp2 wp3 ...  - waypoint
  robot - vehicle
  cup01 ... - interactable
  box01 ... - box
  block01 ... - block
)
(:init
  (can-pickup robot cup01)
  (can-push robot block01)
  (can-push robot cup01)
  (can-stack-on box01 block01)
  (can-stack-on block01 cup01)
  ...)
(:goal ...)
)
```

Figure 3: A fragment of an example problem from the `tidy-room` domain.

3 Predicting Missing Edges in a Multigraph

In this section we describe the procedure of predicting missing edges in a partially-known multigraph. We use M^3VR to extract similarities among vertices based on known edges and to estimate missing edges, embodying the process of associative learning.

M^3VR is a maximum-margin learning framework for predicting incomplete data. The main idea is to capture the hidden structure in the graph. The structure of the graph represents the underlying structure of the environment. Depending on the regularities which occur in environments, these structures can be less and higher complex. In the example in figure 1 it is unknown if `(robot can-pickup block01)`. A prediction is made based on the available information about the `robot` and `block01`. By observing existing relations, one can recognize that `block01` and `cup01` are similar. Thus it is predicted that `robot can-pickup block01` is *true*.

Generalized, the problem of predicting missing relations is a problem of predicting directed edges from the vertices in a set B to the vertices in a set U. We reconstruct a function $f : B \times U \rightarrow E$ from knowledge about existing edges. The function f describes the mapping of vertices to a complete set of edges E. In this way, the set of edges E', representing the known propositions, can be used to measure the similarity between the elements of the vertex set V representing the objects. Using this measure of similarity, missing edges, representing unknown predicates, can be predicted.

For an origin vertex b and a destination vertex u, we define the vector of edges

$$\mathbf{e}_{bu} = \{L_{pred}(b, u) | pred \in Ps\}$$

For example, in the graph given in Figure 1 there will be the

vector:

$$\begin{aligned}&\mathbf{e}_{robot,block01}\\&= [L_{can-fit-inside}(robot, block01),\\&\quad L_{can-stack-on}(robot, block01),\\&\quad L_{can-push}(robot, block01),\\&\quad L_{can-pickup}(robot, block01)]\\&= [0, 0, 1, ?]\end{aligned}$$

Then, we define the projections of known edges E' into a set containing origin vertices B and destination vertices U by

$$B' = \{b \in B | \exists pred \in Ps : \forall U, L_{pred}(b, u) \neq ?\}$$

and

$$U' = \{u \in U | \exists pred \in Ps : \forall B, L_{pred}(b, u) \neq ?\}$$

Finally, the learning problem is given by a set of sample items consisting of three elements $(b, u, \mathbf{e}_{bu})$, where $(b, u) \in B' \times U'$, and $\mathbf{e}_{bu} \in E'$. To realize this learning task we set up an optimization routine for maximum-margin regression.

Since the function f that defines how edges are assigned to vertices can be very complex, a transformation is applied to both vertices and edges, embedding them into Hilbert spaces[1]. We assume that:

Condition 1 *There exists a mapping ϕ from V into a Hilbert space H_ϕ, with the kernel function κ_{vertex} defined on all possible pairs B' and U' of all subsets of $B \times U$ such that $\kappa_{\text{vertex}}(B', U') = \langle \phi(B'), \phi(U') \rangle$.*

Condition 2 *Similarly there is another mapping ψ of E into a Hilbert space H_ψ with a kernel function κ_{edge} defined for all pairs $e_1, e_2 \in E$ such that $\kappa_{\text{edge}}(e_1, e_2) = \langle \psi(e_1), \psi(e_2) \rangle$.*

H_ϕ and H_ψ are feature representations of the domains of B, U, and E. The vectors $\phi(\cdot)$ and $\psi(\cdot)$ are called *feature vectors*. This allows us to use the inner product as a measure of similarity.

The mapping function can now instead be defined on feature vectors, i.e., $F : H_\phi \to H_\psi$. It is indirectly and partially given by the subset E'. The input for the mapping function F is given by the feature vectors of vertices, and the output is a feature vector of edges. To each $b \in B$ representing an origin vertex, we assign such a mapping F_b .

Reconstructing these mappings is done by finding a vector-valued function that extends E' to all possible pairs of vertices by exploiting the latent interactions between the edges. For each element b we assign a linear operator $\mathbf{W}_b$ which maps H_ϕ to H_ψ. Thus predictor functions can be created as

$$\psi(\mathbf{e}_{bu}) \leftarrow \mathbf{W}_b \phi(u), (b, u) \subset B \cap U \tag{1}$$

$\mathbf{W}_b$ is a tensor representing a linear transformation projecting elements of H_ϕ into H_ψ, which needs to be learned. The correlation between the vectors $\mathbf{W}_b \phi(u)$ and $\psi(\mathbf{e}_{bu})$ is described by the inner product $\langle \psi(\mathbf{e}_{bu}), \mathbf{W}_b \phi(u) \rangle_{H_\psi}$. If the correlation between $\mathbf{W}_b \phi(u)$ and $\psi(\mathbf{e}_{bu})$ is higher, the inner product will have a greater value. As a consequence $\psi(\mathbf{e}_{bu})$ can be predicted by a linear function $\mathbf{W}_b \phi(u)$.

Known edges are used to determine $\mathbf{W}_b$ for each mapping. A learner is assigned to learn each mapping. The number of learners is equal to the number of vertices. To exploit the knowledge about existing edges linking different vertices of B, learners are coupled into one assembly by shared slack variables representing the loss to be minimized by the learners with respect to constraints. Detailed descriptions of the optimization procedure can be found in related work (Krivic et al. 2015; Ghazanfar, Prügel-Bennett, and Szedmak 2012). In this procedure there are as many constraints as the number of known edges in E'. Therefore the complexity of the prediction problem is equal to $O(|E'|)$, where $|E'|$ stands for the cardinality of the set E'.

Once determined, the linear mappings $\mathbf{W}_b$ allow us to make predictions of missing edges for elements b. The value of the inner product of the edge feature vector $\psi(\mathbf{e}_{bu})$ and $\mathbf{W}_b \phi(u)$ can be seen as a measure of confidence in that edge belonging to a specific class (in this case 0 or 1):

$$\begin{aligned}\text{conf}\{L^*_{pred}(b, u) = c\} =&\\ \langle \psi(\mathbf{e}_{bu} | L^*_{pred}(b, u) = c), \mathbf{W}_b \phi(u) \rangle_{H_\psi}&\end{aligned}$$

where $c \in \{0, 1\}$, and $L^*_{pred}(b, u)$ is an unknown value in $\mathbf{e}_{bu} \in E \setminus E'$ of predicate $pred$.

For each prediction $L^*_{pred}(b, u)$, we update the existence of the directed edges:

$$L_{pred}(b, u) = \arg\max_{c \in \{0,1\}} \text{conf}\{L^*_{pred}(b, u) = c\}$$

Thus, the graph is completed.

Extending a Partial State with Predictions

Given a complete multigraph, we extend the partially-known initial state $init$ by adding literals to the state:

$$(l_p \in \textit{init}) \leftrightarrow (L_{pred}(b, u) = 1)$$

where l_p is the positive literal of proposition p, formed from predicate $pred$ between objects b and u. For example, Figure 1 contains an edge representing the proposition `can-pickup(robot, block01)`.

Initially, $L_{pickup}(robot, block01) = ?$. After prediction, $L_{pickup}(robot, block01) = 1$. Therefore, we add to the initial state the literal `(can_pickup robot block01)`.

4 Evaluation

To evaluate the approach, we integrated the prediction into a planning and execution framework, ROSPlan (Cashmore et al. 2015). With this framework we were able to generate randomised problem instances from four domains: *tidy-room*, inspired by the problem of cleaning a child's room with an autonomous robot presented in Krivic et al. (2015), in which actions for manipulating objects have been added to the robot's capabilities; *course-advisor*, adapted from Guerin et al. (2012); *mars-rovers*, a multi-robot version of the navigation problem of Cassandra et al. (1996), in which several

[1]Hilbert spaces are high dimensional vector spaces with inner product as scalar quantity associated to the each pair of vectors.

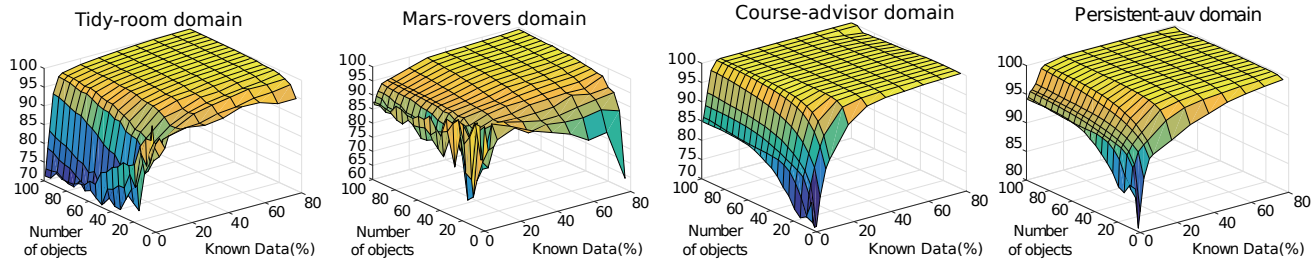

Figure 4: Accuracy results of 10-fold cross validation obtained by variating number of objects and the known data percentage.

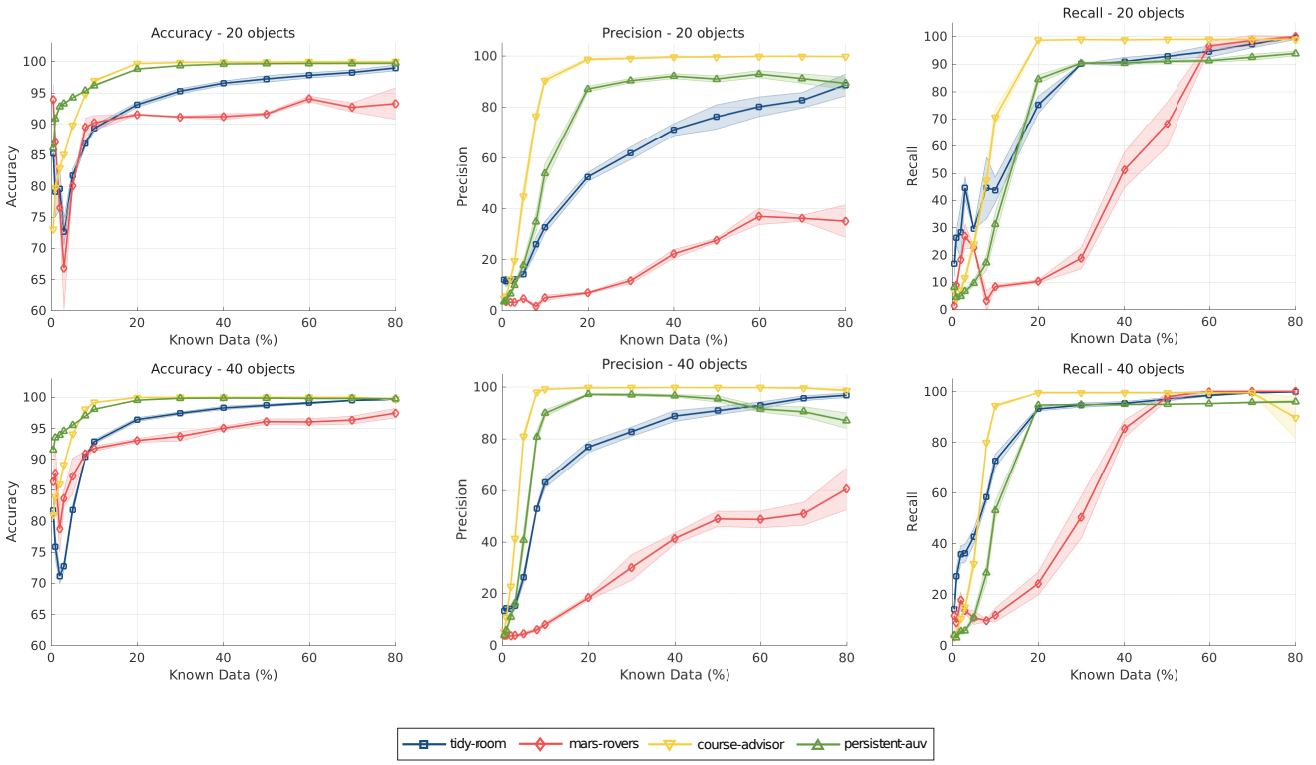

Figure 5: Comparison of accuracy, precision and recall with standard deviation for all 4 domains in case of 20 and 40 objects. Each of the six graphs represents one slice through each of the four graphs of Figure 4, together with standard deviations.

robots are gathering samples; and *persistent-auv*, described by Palomeras et al. (2016).

In each domain we generate initial states, varying the size of the problem and percentage of initial knowledge. The size is varied by increasing the number of objects from 5 to 100 by an increment of 5. This increases the number of propositions quadratically. The initial knowledge is varied by generating complete initial states and removing literals at random. The percentage of initial knowledge was varied between 0.5% and 80% (14 values). For each combination of parameters we randomly created 10 instances of a problem utilizing 10-fold cross validation.

The prediction was applied to every problem. The result of the prediction was compared to the ground truth. Figure 4 shows the mean accuracy in each domain. The accuracy is the number of truly predicted relations as a percentge of all missing relations.

The number of learned relations is large for each domain: for a small percentage of initial knowledge (20%) the overall accuracy of the process is higher than 90% for most of the problems except for the rovers domain with a number of objects less than 20 (Figure 6). Accuracy increases with the size of the problems. With 40 objects in the each problem domain already 8% of known data is enough for accuracy over 90%. *Course-advisor* and *persistent-auv* domains contain more instances of objects and more predicates compared with the other two domains. This results in larger networks. Thus, for these domains, the accuracy is better for smaller numbers of objects as well.

Since all four domains contain more negative than posi-

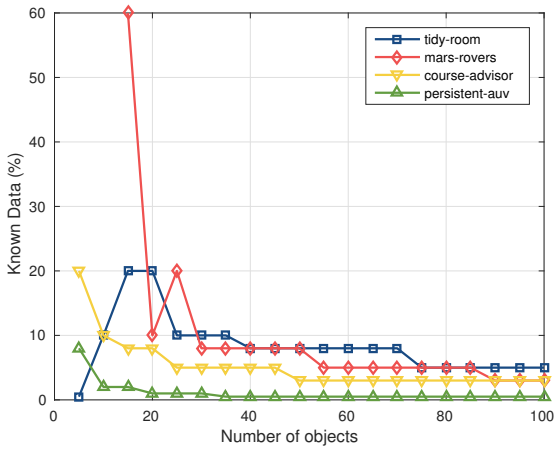

Figure 6: Minimal percentage of known data which gives stable accuracy equal to or higher than 90%.

no. of objects	time to solve conting.	time to solve predict.	plan duration (s) conting.	plan duration (s) predict.
2	2.57	2.00	210.00	160.00
3	19.55	2.03	320.00	290.00
4	94.34	2.11	450.00	440.00
5	346.99	2.25	580.00	510.00
6	807.28	2.37	670.00	590.00
7	-	2.61	-	660.00

Table 1: Average time to solve (mean in seconds) and average plan duration for problems in the *tidy-room* domain, with varying numbers of objects. Times are for contingency planning and planning after prediction. Initial knowledge was 30%. Times in the prediction column include prediction time. Planners were given a time limit of 1800 seconds.

tive literals (the average ratio is 76:1) we also analysed precision and recall for the category representing positive literals. Precision is the percentage of all literals predicted to be positive which are predicted correctly. Recall is the percentage of positive literals in the ground truth which were correctly predicted.

We also give a comparison for each domain for 20 and 40 objects in Figure 5. The *course-advisor* and *persistent-auv* domains appear to be simpler than the *tidy-room* and *mars-rovers* domains and accuracy is high with a very small proportion of known data. However in the *course-advisor* domain overfitting effects appear for high percentages of known data and many objects (Figures 4 and 5).

With 20% of the initial predicates and with 20 objects recall and accuracy are higher than 90% for all domains except *mars-rovers*. To examine the problem size conditions we extracted from the testing results which amount of the known data is needed to achieve accuracy higher than 90%. This is shown in the Figure 6. Accuracy increases with the size of the problems. With 40 objects in the each problem domain already 8% of known data is enough for accuracy over 90%. *Course-advisor* and *persistent-auv* domains contain more instances of objects and more predicates compared with the other two domains. This results in larger networks. Thus, for these domains, the accuracy is better for smaller numbers of objects as well.The accuracy depends on the structure of used domains, their size and complexity of regularities in relations between objects. Results show good predictions for large and small problems with different complexities.

This combination of high accuracy, precision and recall allows us to solve many otherwise unsolvable instances, while maintaining a high degree of robustness. Without preprocessing, no problems could be solved.

To analyse the effect of prediction in regard to the planning state space, we count the number of reachable actions before and after prediction. Figure 7 shows the the number of newly reachable actions as a fraction of the total number of actions in the ground truth. The increase in reachable actions is very large, even for a small amount of initial knowledge. These results show that after prediction, the number of all reachable actions was very high in all experiments, almost the same as the number of actions enabled by ground truth. This increase in valid actions enabled by the prediction demonstrates an increase in size of reachable state space.

The prediction approach is not mutually exclusive to approaches to contingency and conformant planning approaches. In fact, these approaches can be usefully combined, as we discuss in the conclusion. However, we compare against a purely contingent planning approach in order to illustrate the benefit in complexity. Sensing actions

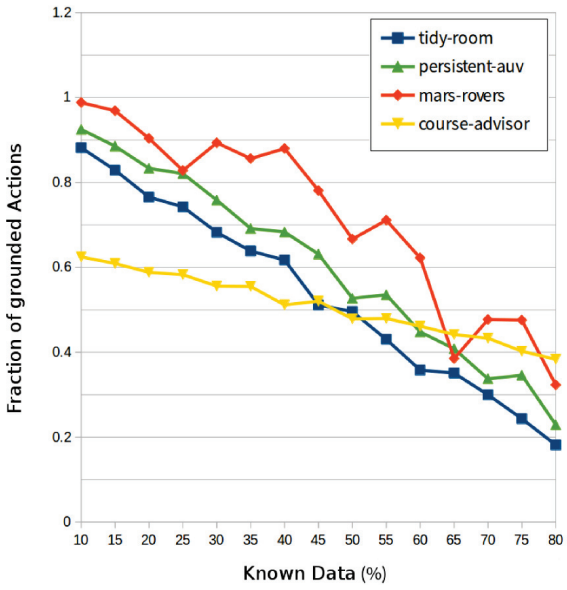

Figure 7: Number of newly reachable actions after prediction divided by reachable actions in the ground truth. For problems with 20 objects and varying amounts of initial knowledge.

were introduced into the *tidy-room* domain, allowing the agent(s) to determine the ground truth of unknown propositions. We used the planner CLG (Albore and Geffner 2009) to solve problems in these extended domains, recording the time taken to solve and the duration of the execution trace of the contingent plan. These data are shown in Table 1.

Each problem was generated in the same way as described above, with 30% initial knowledge. We compare these times against the time taken to solve the problems by first applying prediction and then solving using the planner POPF (Coles et al. 2010), and validating against the ground truth using VAL (Fox 2004). The times for POPF include the time taken to perform the prediction. With 30% initial knowledge all plans produced by POPF after prediction were valid.

These results show that while both approaches produce valid plans given 30% initial knowledge, a prediction approach scales far better. By pre-processing the problem we are able to avoid complexity at small cost to robustness, particularly when the initial knowledge is above 30%.

The plan duration shows a clear benefit of initial state prediction. This improvement comes from the quality of the plans produced by the respective planners, but also from the need of sensing actions in the contingent plan, which increase the duration by an average 31 seconds.

5 Conclusion

We have shown how an initial state with uncertainty can be represented as a partially-known multigraph, how the M^3VR framework can be used to predict edges in such a graph, and how these edges can then be reintroduced into the initial state as predicted propositions.

Learning a world model can be improved with exploitation of the obtained knowledge in the learning process itself. An agent can build hypotheses on unknown relations in the world model by associating among the existing and possible relations.

This approach is performed offline and is not an execution strategy in itself. It is orthogonal to online approaches in dealing with uncertainty in the initial state, and can be combined. Moreover, while we compared with contingency planning, these approaches are not exclusive.

In future work we intend to investigate how predictions can be used in order to inform a contingent planning approach in two ways:

1. By using the confidence values of predictions, filtering the number of unknown facts verifiable by sensing action. A confidence threshold indicates a likely fact that should be verified by a sensing action, while a high confidence can be assumed true.
2. Directing the executive agent to perform sensing actions that, while not immediately supporting actions leading towards the goal, will allow for a higher confidence prediction of many other facts involving similar objects.

Without integration into a more sophisticated execution strategy, our evaluation has shown that this approach accurately predicts a surprisingly large number of facts in structured domains, even with few objects.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 610532, SQUIRREL.

References

Albore, A.; Palacios, H., and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, 52–61.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *Proceedings of the 22nd International joint conference on Artificial Intelligence (IJCAI'11)*.

Bonet, B. 2009. Deterministic POMDPs revisited. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*, 5966.

Brafman, R. I., and Shani, G. 2014. Replanning in domains with partial information and sensing actions. *CoRR*.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. Rosplan: Planning in the robot operating system. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*.

Cassandra, A. R.; Kaelbling, L. P.; and Kurien, J. A. 1996. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of the 20rd International Conference on Automated Planning and Scheduling (ICAPS'10)*, 42–49.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Res. (JAIR)* 20:61–124.

Fox, R. H. D. L. M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*.

Ghazanfar, M. A.; Prügel-Bennett, A.; and Szedmak, S. 2012. Kernel-mapping recommender system algorithms. *Information Sciences* 208:81–104.

Guerin, J. T.; Hanna, J. P.; Ferland, L.; Mattei, N.; and Goldsmith, J. 2012. The academic advising planning domain. In *Proceedings of the 3rd Workshop on the International Planning Competition at ICAPS*, 1–5.

Hill, W. F. 1984. Conditioning and associative learning. *The American Journal of Psychology* 97(3):472–474.

Hoffmann, J., and Brafman, R. I. 2005. Contingent planning via heuristic forward search witn implicit belief states. In *Proceedings of the 15th International Conference on AutomatedPlanning and Scheduling (ICAPS'05)*, 71–80.

Krivic, S.; Szedmak, S.; Xiong, H.; and Piater, J. 2015. Learning missing edges via kernels in partially-known graphs. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Muise, C. J.; Belle, V.; and McIlraith, S. A. 2014. Computing contingent plans via fully observable non-deterministic planning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2322–2329.

Palacios, H., and Geffner, H. 2006. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proceedings of the 21st Conference on Artificial Intelligence (AAAI'06)*.

Palomeras, N.; Carrera, A.; Hurts, N.; Karras, G. C.; Bechlioulis, C. P.; Cashmore, M.; Magazzeni, D.; Long, D.; Fox, M.; Kyriakopoulos, K. J.; Kormushev, P.; Salvi, J.; and Carreras, M. 2016. Toward persistent autonomous intervention in a subsea panel. *Autonomous Robots*.

Pavlov, I., and Anrep, G. 2003. *Conditioned Reflexes*. Dover Publications.

Sammy Davis-Mendelow, Jorge A. Baier, S. A. M. 2013. Assumption-based planning: Generating plans and explanations under incomplete knowledge. In *Proceedings of the 27th conference on Artificial Intelligence (AAAI'13)*, 209–216.

Smith, D. E., and Weld, D. S. 1998. Conformant graphplan. In *Paper presented at the meeting of the AAAI/IAAI (AAAI'98)*, 889–896.

Szedmak, S.; Ugur, E.; and Piater, J. 2014. Knowledge propagation and relation learning for predicting action effects. In *Intelligent Robots and Systems (IROS'14)*, 623–629.